

UNITED STATES PATENT APPLICATION

FOR

**SYSTEM AND METHOD FOR  
INTEGRATING MEDIA OBJECTS**

**Inventors:**

**Xiaochun Nie  
Christopher L. Flick**

Prepared By:

**BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN**  
12400 Wilshire Blvd., 7th Floor  
Los Angeles, California 90025-1026  
(310) 207-3800

10022151.121401

## **SYSTEM AND METHOD FOR INTEGRATING MEDIA OBJECTS**

### **BACKGROUND**

#### **Field of the Invention**

[0001] This invention relates to the field of application program interfaces for supporting media objects. More specifically, the invention relates to a system and method for integrating media objects for viewing on and manipulation via a computing device such as a personal computer.

#### **Background**

[0002] Computers and other processor based devices handle many kinds of data. Various kinds of data may be accessed by computers, including two-dimensional (2D) objects and 2D virtual reality enhanced images having three-dimensional (3D) properties. Operating systems provide utilities which may be used by application programs for various purposes, including the low-level handling of media objects. Many operating systems do not provide more than rudimentary support for the handling of 2D media objects and do not generally support enhanced 2D media objects. Additional utilities may be provided via application program interfaces commonly referred to as APIs. Some extensions to operating systems may provide the facility to separately view various kinds of 2D media objects. One such operating system extension is the QuickTime® System from Apple Computer, Inc. of Cupertino, California. The ability to integrate multiple 2D media objects and 3D enhanced 2D objects into a scene having 3D properties and to then manipulate the scene and the objects is not provided in current operating systems and operating systems extensions.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0003] The invention described herein is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0004] **Figure 1** illustrates a personal computing device and environment in which the embodiments of the method and system described herein may be implemented.

[0005] **Figure 2** illustrates the flow of actions taken according to an embodiment of a method to integrate media objects into a scene.

[0006] **Figure 3** illustrates a three-dimensional graph showing the effects of rotating an object or scene about each of three axes.

[0007] **Figure 4A** illustrates a camera and its coordinate system in relation to a subject and its image frame.

[0008] **Figure 4B** illustrates an object coordinate system superimposed in a scene coordinate system.

[0009] **Figure 5A** illustrates a first object.

[0010] **Figure 5B** illustrates a second object.

[0011] **Figure 5C** illustrates an example viewable scene that includes a first object integrated with a second object.

[0012] **Figure 5D** illustrates an example viewable scene that includes a first object integrated with a second object after the field of view of the scene has been by zooming in.

[0013] **Figure 5E** illustrates an example viewable scene that includes a first object integrated with a second object after the scene has been panned to the left.

[0014] **Figure 5F** illustrates an example viewable scene that includes a first object integrated with a second object after the scene has been panned to the right.

### **DETAILED DESCRIPTION**

[0015] The invention described herein provides various embodiments of a method and system for integrating two-dimensional (2D) and three-dimensional (3D)

media objects for viewing on and manipulation via a computing device such as a personal computer. As many operating systems provide only rudimentary support for handling 2D and 3D media, the various embodiments of the system and method described herein may be implemented as an application program interface (API) which allows application programs to take advantage of the features of the method and system for integrating 2D and 3D media objects described herein. The various embodiments of the system and method described herein may be incorporated as part of an operating system or provided as an extension of an operating system which exists between the operating system and application programs.

**[0016]** **Figure 1** illustrates a personal computing device and environment in which the embodiments of the method and system described herein may be implemented. Computing device 100 may be any computing device that can execute software programs, including, but not limited to, personal computers, servers, cellular telephones, computer tablets, personal digital assistants, portable computers, computer workstations, etc. Computing device 100 may comprise a processor 110 to execute software programs. Processor 110 may be any computer processor known to those skilled in the art. Although only one processor is shown, two or more processors may be included in computing device 100. When executing programs, processor 100 utilizes memory 112. Memory 112 may be any form of random access memory (RAM) known to those skilled in the art. Information is read from and written to storage device 120. Storage device 120 may be any kind of machine readable medium including, for example, magnetic media such as disk drives and magnetic tape; optical drives such as compact disk read only memory (CD-ROM) and readable and writable compact disks (CD-RW); stick and card memory devices; ROM, RAM, flash memory devices and the like; whether internal, such as storage device 120 and, in other embodiments, directly coupled, accessible locally or remotely via network 170, and accessible via electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.) via network access device 162. In various embodiments, network access device 162 may be a modem, network interface unit, or other communications device that allows for communication with other computing devices, such as remote computing device 180.

**[0017]** Personal computer 100 may include graphics controller 132 that allows for the display of information such as text and graphics on display monitor 134. Graphics controller 132 may be any graphics controller known to those skilled in the art, and display 134 may be any display monitor known to those skilled in the art, including, for example, a cathode ray tube (CRT) display monitor and thin film transistor (TFT) display screen. Personal computer 100 may also include audio controller 136 to which speakers 138 may be coupled. Personal computer 100 may play audio including music, voice, and synthesized sound to a user of the personal computer via audio controller 136 and speakers 138. Audio controller 136 may be any audio controller that may be compatible with the Sound Blaster® *de facto* standard popularized by Creative Technology Ltd. of Singapore and that may support 3D audio. A user may access computing device 100 via any input device known to those skilled in the art, such as, for example, keyboard 144, mouse 146, track ball (not shown), keypad (not shown), pen and tablet (not shown), etc. These input devices may be coupled to computing device 100 by input/output (I/O) controller 142. In one embodiment, I/O controller 142 may be a universal serial bus (USB) controller and the input devices may be USB enabled.

**[0018]** Processor 110, memory 112, storage device 120, graphics controller 132, I/O controller 142 and network access device 162 may be coupled to one another via and communicate with one another over bus 150. Bus 150 may be any bus known to those skilled in the art. Although only one bus is depicted, multiple buses may be used in computing device 100. In addition, other components and controllers known to those skilled in the art (not depicted) or multiple instances of depicted components and controllers may be included in computing device 100.

**[0019]** The methods described herein may be implemented as software and stored for execution on storage device 120 of computing device 100 as media processing software (MPS) 124. In one embodiment, the MPS may be a suite or group of software functions, programs, routines, etc. that provide data structure definitions and serve as an API to application programmers. When executing, the MPS may rely on and utilize functions and features made available by operating system 122. Operating system 122 may be stored on storage device 120 or another local or remote

storage device. Operating system 122 may be any well known or proprietary operating system, such as, for example, Mac OS X available from Apple Computer, Inc. of Cupertino, California. In another embodiment, the MPS may be incorporated as part of an operating system. In other embodiments, the MPS may exist as hardware or as a combination of hardware and software. In one embodiment, the system and method described herein rely on and have available to them an existing set of support tools in the form of a library of functions comprising an API that provides for the manipulating, rendering and display of various 2D and 3D objects and, in some embodiments, for the rendering of stereo and/or 3D audio. An example of such a set of APIs and tools are those available in the QuickTime® 5 cross-platform multimedia software developer's toolkit available from Apple Computer, Inc. of Cupertino, California, such as API tool set may support the OpenGL graphics API. (See The OpenGL® Graphics System: A Specification, version 1.3, available from Silicon Graphics, Inc. of Mountain View, California.)

**[0020]** In various embodiments, the system and method described herein allow for the integrating, compositing or combining of two or more kinds of 2D and 3D objects into a scene. The scene is said to include various scene objects. In one embodiment, a soundtrack may be associated with a scene. In another embodiment, a soundtrack may be included with an object such that when an object is activated by a user such as by a mouse click, the soundtrack is played. In one embodiment, when an object in a scene is manipulated by a user, such as by dragging the object via a mouse, the location of the audio associated with the object may be changed to reflect the manipulation and position in the 3D scene. That is, for example, the audio may be rendered to be played from left to center to right as an object is dragged across a scene. In these embodiments, the soundtrack may be stored according to any well-known format such as, for example, the audio interchange file format (AIFF), Moving Picture Experts Group audio level 3 (MP3) format, and wave format (WAV). In these embodiments, the MPS may play the soundtrack by invoking a library function, an operating system utility, or a stand-alone audio application program.

**[0021]** The location and perspective from which a user may view the scene may be defined by what is referred to as a scene camera and may also referred to as a

camera view. In one embodiment, multiple scene cameras for a scene may be supported such that a user may switch back and forth between two, and in other embodiments, more, scene cameras. In various embodiments, there may be multiple and varied supported 2D and 3D objects. In various embodiments, multiple formats of 2D and 3D objects may be supported including, for example, OpenGL3D objects. An example of a 2D media object is a bitmap image which, in one embodiment, may be known as a sprite. In one embodiment, the source for a sprite can be a single bitmap, a bitmap image sequence, a vector image, a video track, a live stream, or a source specified by a uniform resource locator (URL). Examples of 3D media objects include object movies and OpenGL 3D objects. Object movies are not truly movies and are not truly 3D, but they are an enhanced form of 2D having some 3D properties. Although not truly 3D entities, object movies and other similarly enhanced 2D objects will be referred to as 3D objects and as enhanced 2D objects herein. Object movies are created by photographing or rendering a series of views of the subject at carefully spaced angles of tilt and rotation and assembling the images. In one embodiment, each view of the subject in an object movie is a still image. In another embodiment, one or more views of the subject of an object movie may be a video clip or animation. In one embodiment, an object movie may be created by photographing a series of images of an object as the object is rotated on a platter along different axes. In another embodiment, the series of images comprising an object movie may be captured when rotating a camera from around the object along different axes from a fixed distance.

**[0022]** **Figure 2** illustrates the flow of actions taken according to an embodiment of a method to integrate 2D and 3D objects. In practice, after a computing device is up and running the MPS will register with the operating system according to well-known methods. The MPS may receive a request to create a new scene, as shown in block 210. This request is in the form of a function call from an application program written according to the MPS API. The MPS then receives a request to add an object to the scene, as shown in block 212. Upon receiving the request, depending on the type of object, the MPS processes the object to create a scene object, including adding 3D properties to the scene object and setting the orientation and translation of the scene object with respect to the scene, as shown in block 214. If there are more objects to be added to the scene, as shown in block 216, blocks 212 and 214 are repeated. After all

objects to be included in the scene have been added to the scene, the MPS may receive a request to build a camera view of the scene and may build the requested camera view, as shown in block 218. If more camera views are to be added for the scene, as shown in block 220, block 218 is repeated until no further camera views are to be added and built. The opening scene will then be rendered and displayed to a user by the MPS invoking an application supplied callback routine and/or calling one or more appropriate library functions, as shown in block 222. In one embodiment, the opening scene may be set by a special API, may be set by a variable or argument passed with the create scene API, or may be system chosen according to default requirements. In one embodiment, the MPS invokes the callback routine which in turn calls one or more library functions. In another embodiment, the MPS may directly call the library functions, and may then invoke the callback routine.

**[0023]** To achieve the functionality described in the prior paragraph, the scene manager maintains pertinent data about each object, the relationship of each object to the other objects, and the relationship of each object to the scene, including layering data. It is the layering data which may be used to determine which objects are in front of other objects, and consequently, which objects are background and foreground. In various embodiments, each object may also include opacity data. In one embodiment, opacity data may be included in an alpha channel associated with the objects. Various degrees of opacity may be supported, including, for example, fully transparent, half transparent, and varying degrees of transparency. In one embodiment, the MPS calls library functions to display each of the objects in the scene. In doing so, the MPS maintains the position of each object relative to the scene and maintains the camera view relative to the scene. Although referred to as a camera view, the camera view or scene camera is actually the point and perspective from which a user will view a display of the scene.

**[0024]** In one embodiment, as part of displaying the scene, the user is provided the opportunity to control the view of the scene such as by panning to the left and right, tilting up and down, and zooming in and out. The user may also, in some embodiments, be able to change the point of view, also referred to as the point of reference. The MPS may receive user input, which may, in one embodiment, be in the



10022151-121401

form of mouse movement and mouse click as well as keyboard input, changing the camera view, changing the object selected, manipulating the object, manipulating the scene, and/or manipulating the scene camera, as shown in block 224. As to object selection, in some embodiments, the user may be provided the ability to select which of the objects in the scene should be a current primary object or focal point. Upon receiving user input, the MPS then invokes an application program supplied callback routine and/or calls one or more appropriate library functions to achieve the user requested change or manipulation and to render and display the resulting scene, as shown in block 226. In one embodiment, calls to the library functions may be made by the callback routine, and, in other embodiments, may be made by the application program in response to the callback routine. In another embodiment, the MPS may call one or more appropriate library functions, and may then invoke the callback routine. The user may continually navigate within a scene to view as much as the scene as is desired. As such, flow then continues at block 224, and may continue in a loop until the program is exited.

**[0025]** **Figure 3** illustrates a three-dimensional graph showing the effects of rotating an object or scene about each of three axes. Each of the objects which may be combined into a scene has its own object coordinate system in 3D space which is translated into the scene coordinate system by a translation vector. Traversing a scene such as by panning, or changing a user's view of the scene to the left or the right, will result in corresponding panning of an object contained in the scene including changes to the position and orientation of the object with respect to the scene. Panning requires that the scene be rotated about the scene's y-axis, as shown by reference number 310. When a user pans in a scene, each object or the image frames representing each object are correspondingly panned by the MPS by calling a function or tool provided by a prior art library of APIs to rotate the appropriate image frame. The input to the library function is based on the translation and orientation of the scene object with the scene according to the frame vector and the user input such that the scene is rotated along the scene's y-axis. If the object to be rotated is an object movie or other enhanced 2D object, the library function may retrieve the appropriate image frame of the object which was captured at a corresponding orientation that matches the orientation to where the user panned in relation to the scene.

**[0026]** Generally, tilting a scene or changing a user's view of a scene up or down, requires that the scene be rotated about the scene's x-axis, as shown by reference number 320. When a scene is to be tilted, a function or tool provided by a prior art library of APIs is called to rotate each of the objects based on the translation vector and the amount of the rotation along the scene's x-axis. In one embodiment, this may be accomplished by retrieving an appropriate image frame of the object at a corresponding orientation. Similarly, in some embodiments, support for yaw may also be provided. Changing the yaw of a scene, or changing a user's view in relation to a vertical axis, requires that the scene be rotated about the scene's y-axis, as shown by reference number 330. To rotate a scene, a function or tool provided by a prior art library of APIs is called to rotate each of the scene objects based on their translation vectors and the user requested rotation of the scene about the scene's z-axis. Depending on the object and the embodiment, the rotation may be achieved by application of a software algorithm or by retrieving an appropriate image frame captured corresponding to the desired orientation.

**[0027]** If the user requests that the field of view of the scene be changed by zooming in or out, the appropriate library function is called regarding each of the objects in the scene such that each of the objects is zoomed according to the user input, the translation data for each object, and the scene data. In one embodiment, the zooming may be achieved wholly algorithmically. In another embodiment, the zooming may be achieved by retrieving an appropriate image frame at a desired orientation and field of view.

**[0028]** **Figure 4A** illustrates a camera and its coordinate system in relation to a subject and its image frame. In one embodiment, some objects in a scene may initially be captured as a series or sequence of image frames taken by a real camera of the subject that is represented as an object. Either the real camera may be rotated about the subject, or the subject may be rotated in front of the real camera to obtain the sequence of images needed to comprise the object. In this embodiment, when capturing the sequences of images of an object, the definition of the image frame is a 3x3 rotation matrix and a 3D translation vector based on the relationship between the image frame of the subject and the camera. The 3x3 rotation matrix defining subject 400 maps the

orientation of image frame 401 as an object according to object x-axis 402, object y-axis 403, and object z-axis 404, and the 3D translation vector defines the position of the object in the camera coordinate system. The camera 406 obtains the image frame 401 of subject 400 from its own coordinate system having camera x-axis 407, camera y-axis 408 and camera z-axis 409.

**[0029]** Figure 4B illustrates an object coordinate system superimposed on a scene coordinate system. Each object in a scene is defined as a series or sequence of image frames and orientation data. These image frames may be retrieved, accessed and manipulated by available library functions accessing the media object. Each object is initially defined in its own coordinate system. However, when creating a scene, a scene coordinate system is created. To exist in a scene, each object added to the scene must be placed at a location in the scene. The definition of where each object is to exist in a scene is determined by a translation vector which is used in conjunction with the object's coordinate system. For example, each object's position and orientation may be defined by a 3x3 matrix representing object axes 410, 412 and 414, as shown in Figure 4B and a translation vector. The translation vector directs where to place the object in relation to scene axes 422, 424 and 426, as shown in Figure 4B. The 3x3 rotation matrix is combined with the translation vector to form a 4x4 matrix known as a scene object.

**[0030]** More specifically, the 3D translation vector may be defined as  
 $T = [x, y, z]$ . The rotation transformation of T through rotation matrix R is:

$$\begin{bmatrix} xx \\ yy \\ zz \end{bmatrix} = \begin{bmatrix} R1 & R2 & R3 \\ R4 & R5 & R6 \\ R7 & R8 & R9 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R1*x + R2*y + R3*z \\ R4*x + R5*y + R6*z \\ R7*x + R8*y + R9*z \end{bmatrix}$$

**[0031]** Transforming a scene object in the form of a 4x4 matrix may be viewed as:

$$\begin{bmatrix} xx \\ yy \\ zz \\ 1 \end{bmatrix} = \begin{bmatrix} R1 & R2 & R3 & T1 \\ R4 & R5 & R6 & T2 \\ R7 & R8 & R9 & T3 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R1*x + R2*y + R3*z + T1*x \\ R4*x + R5*y + R6*z + T2*y \\ R7*x + R8*y + R9*z + T3*z \\ 1 \end{bmatrix}$$

[0032] This is mathematically a definition of the linear transformation. The geometrical meaning of a rotation matrix may be considered as representing three vectors such that:

$$R = \begin{bmatrix} | & |R1| & |R2| & |R3| & | \\ | & |R4| & |R5| & |R6| & | \\ | & |R7| & |R8| & |R9| & | \end{bmatrix} = \begin{bmatrix} | & Vx & Vy & Vz & | \end{bmatrix}$$

[0033] Transformation of the above rotation matrix results to a weighted sum of the three axes in 3D space. The three axes Vx, Vy, Vz can be of any length and orientation, and they do not have to be perpendicular to each other.

$$\begin{bmatrix} | & xx & | \\ | & yy & | \\ | & zz & | \end{bmatrix} = \begin{bmatrix} | & R1 & R2 & R3 & | \\ | & R4 & R5 & R6 & | \\ | & R7 & R8 & R9 & | \end{bmatrix} * \begin{bmatrix} | & x & | \\ | & y & | \\ | & z & | \end{bmatrix} = \begin{bmatrix} | & Vx*x & + & Vy*y & + & Vz*z & | \end{bmatrix}$$

[0034] **Figure 5A** illustrates a first object, Roman bust 500. Roman bust 500 is an enhanced 2D image having 3D properties known, in one embodiment, as an object movie, a series of captured photographic images taken at various prescribed orientations. **Figure 5B** illustrates a second object, school room 510. School room 510 is also an object movie. By using the MPS API described herein, a scene may be created by combining these objects. **Figure 5C** illustrates an example viewable scene that includes a first object, Roman bust 500, integrated with a second object, school room 510. An application program used to create the scene in **Figure 5C** is an authoring tool. The authoring tool is an application program which allows a user to acquire objects and place them into a scene. Authoring tools may rely on the API provided by the MPS to allow a user to create a scene.

[0035] After a scene created with an authoring tool is displayed, a user may provide input to manipulate the user's view of the scene. This may be referred to as manipulating the camera, such as by panning, tilting, translation, and zooming. As set forth above regarding **Figure 2**, when user input is received, the MPS invokes the appropriate call-back routine of the application program that displayed the scene to handle the manipulation request. The application program will then use the API of the MPS to manipulate the objects and the scene to display the manipulated scene

according to the changed camera view. The MPS processes the revised location of each of the objects in relation to the scene and the manipulation request and then uses existing library functions to manipulate each of the objects to an appropriate orientation scene and then to display the scene. For example, **Figure 5D** illustrates an example viewable scene that includes a first object, Roman bust 500, integrated with a second object, school room 510, after the field of view of the scene has been changed by zooming in. **Figure 5E** illustrates an example viewable scene that includes a first object, Roman bust 500, integrated with a second object, school room 510, after the scene has been panned to the left. **Figure 5F** illustrates an example viewable scene that includes a first object, Roman bust 500, integrated with a second object, school room 510, after the scene has been panned to the right.

[0036] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes can be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.